A Semantic Characterisation for Faults in Replicated Systems [1]

Padmanabhan Krishnan

Department of Computer Science
University of Canterbury, Private Bag 4800
Christchurch, New Zealand
Email: paddy@cosc.canterbury.ac.nz

Abstract

We describe a process algebraic approach to the semantics of replicated systems. We extend a subset of CCS with a replication operator to model systems with replicated synchronous majority voting. Based on an operational semantics, we define a bisimulation semantics. As the bisimulation semantics does not characterise fault tolerance we define preorders which introduces a hierarchy of faulty processes and fault tolerant processes. We then show how a similar ordering on modal-μ formulae can characterise the fault preorders.

1 Introduction

The principal feature of fault tolerant robust or safety-critical systems is the ability to cope with hardware or software errors. A fault can be defined to be an unexpected event which causes the system to deviate from its expected specified behaviour. With in the context of reactive systems [22], a fault can be defined to be an unexpected change in the operating environment. Unexpected changes can occur, as all system specifications make certain assumptions of an ideal environment. Robust reactive systems are usually able to operate in non-ideal environments.

The aim of this work is to describe a framework in which fault tolerant systems can be studied. The main aspects in building a fault tolerant system include detection, diagnosis and recovery. Strategies to build fault tolerant systems depend on what is classified as a fault. [6] presents a few categories of faults that could occur in communicating systems. These include *omission* fault or failure to send a message, *addition* fault or generation of an spurious message, *value* fault or sending the wrong value, *state-transition* fault or responding incorrectly to the environment and *crash* failure or the inability to interact with its environment.

Associated with a system is a failure model, which is a specification indicating the corrective action on the occurrence of a fault. The failure model chosen for a particular system depends on its functionality. For example, in a student lab environment, shutting down the lab due to an erroneous file-server would be acceptable while a heart-lung machine should not be shut if a sensor is faulty. Also associated with a fault model is containment, i.e., how to limit the effect of a fault. For example, if backups are available one may shut down a server and activate a backup. If this is done transparently the system as a whole continues to work smoothly.

As there are a large number of techniques to detect faults and to recover from them, it is difficult to address all issues in one paper. Even though there are many techniques, a common strategy to make a system robust, is to replicate it and obtain results via synchronous majority voting [3, 6, 8]. In this paper we consider the effect of omission, value and addition on replicated systems.

As robust systems operate in parallel, we develop a theory for replicated systems in the context of theories of concurrent systems. Process calculi such as ACP [4], CCS [18] and CSP [10] are important formalisms in the description of concurrent systems. A trace semantics with extractor functions for replicated CSP processes has been developed [11]. However they do not consider explicit fault modelling.

In this paper we present a calculus similar to CCS for replicated systems with a notion of fault injection. We develop a bisimulation semantics for the calculus and present a complete axiomatisation. The bisimulation semantics is only concerned with the observable behaviour of the system. As replication affects the behaviour of a system with faults, semantic characterisations of the failure classification using preorders is defined. The preorder is relativised with respect to the correct behaviour and if P is less than Q in the preorder, Q is no more faulty (with respect to the correctness criteria) than P. We also develop a logical characterisation of the preorders using the modal-μ calculus [23].

2 Replication

As in CCS [18] we assume a set of atomic actions Λ with typical elements represented by μ_1, μ_2 etc.

[1] A preliminary version of this paper appeared in [15] and [14]

The syntax for the set of processes is defined as follows.

P ::= **0** | μ·P | (P Ⅱ P) | (P + P) | (P | P)
Sys ::= (P ⊛) | μ·Sys | (Sys + Sys) | (Sys | Sys)

Let $\mathcal{P}_R$ be the set of all finite processes over P and $\mathcal{P}$ to be the set of all finite processes over Sys. As usual, **0** represents the terminated process, · action prefix, + non-deterministic choice and | parallel composition. We have not considered restriction, recursion or relabelling. The main reason for not including restriction is that communication assumes the existence of a bijective map on the set of actions. As faults needs not preserve the bijection, the faulty behaviour with restriction is harder to predict. The reason for not including recursion is that some of our results depend on finite behaviour. More work is necessary to determine if the results can be generalised to finite state processes. In the absence of restriction and recursion, processes with relabelling can rewritten as new processes without relabelling; hence we do not consider relabelling. More details for these decisions are discussed in section 5.

We have introduced two new combinators Ⅱ and ⊛. The Ⅱ combinator indicates 'replication'. We do not require the two processes joined by replication to be identical. For example, in (P Ⅱ Q) P and Q can be very different processes. This allows us to model faulty systems, e.g., P represents correct behaviour while Q represents a faulty behaviour. One could also consider (P Ⅱ P) and study the effects of various fault on its observable behaviour. Intuitively, in (P Ⅱ Q) the processes P and Q decide to exhibit a particular behaviour. Their decisions are combined and the action that receives the majority vote is exhibited.

The purpose of the ⊛ combinator needs some explanation. Looking ahead, we are interested in developing a bisimulation semantics [20] for the new calculus. If one only considered elements of $\mathcal{P}_R$, the resulting relations is not a congruence. For example, the process $(\mu_1 \cdot \mathbf{0} \;\text{Ⅱ}\; \mu_1 \cdot \mathbf{0})$ intuitively behaves as $\mu_1 \cdot \mathbf{0}$ and hence would be related. Now consider the behaviour of the processes in conjunction with $\mu_2 \cdot \mathbf{0}$. While the process $(\mu_1 \cdot \mathbf{0} \;\text{Ⅱ}\; \mu_1 \cdot \mathbf{0} \;\text{Ⅱ}\; \mu_2 \cdot \mathbf{0})$ intuitively behaves as $\mu_1 \cdot \mathbf{0}$, the process $(\mu_1 \cdot \mathbf{0} \;\text{Ⅱ}\; \mu_2 \cdot \mathbf{0})$ does not. To obtain a congruence we 'seal' a process, i.e., disallow it to be executed along with another as a replicated process. This is necessary as no finite replication can be said to be sufficient for all faults. We can conclude that, if a calculus has an explicit replication combinator it is necessary to have a sealing combinator. In a later section we will show that by avoiding an explicit replication combinator and by using multisets of actions instead of actions, the seal combinator can be avoided.

As a notational convenience we shall use **0** instead of (**0** ⊛). We also omit the trailing **0**'s; for example we write μ instead of μ·**0**.

The operational semantics is based on labelled transition systems [21] and consists of two parts, one for $\mathcal{P}_R$ and the other for $\mathcal{P}$. The transitions for elements of $\mathcal{P}_R(\rightarrowtail)$ can be perceived as internal moves (i.e., moves of a replication system to obtaining votes) while the transition rules for the elements of $\mathcal{P}(\longrightarrow)$ defines the observable behaviour. This is similar to the notion of high level and low level transition introduced in [7]. As their concern is decomposition of actions at an implementation level they do not consider voting. In our semantics, actions are atomic for both internal and external transitions.

As an action can receive more than one vote, we use multi-sets to represent the state of the voting machine. Addition of votes and declaring the winning action are defined as follows.

Definition: 1 *If O_1 and O_2 are multisets over Λ, define $O_1 + O_2 = O$ such that $\forall\ \mu \in \Lambda\ O(\mu) = O_1(\mu) + O_2(\mu)$*
Given a multi-set O, Voted_Action *$(O) = \{\ \mu \mid \forall\ \mu_1 \in \Lambda,\ O(\mu) \geq O(\mu_1)\}$*

The internal transition rules are defined in figure 1.

The observable transition rules are given in figure 2. The transition rule for sealing is derived from $\rightarrowtail$.

The operational semantics for ·, + and | are as usual and we have introduced rules for Ⅱ and ⊛. We use the above definitions as the basis for the work described in the rest of the paper.

2.1 Bisimulation

In this section we define and provide a complete axiomatisation of a bisimulation relation. In this paper we focus on an interleaving semantics. These definitions could easily be extended to cover an architecture based semantics [12, 13]. While an architecture based semantics will be useful in studying the effect of hardware failure, in this paper we concentrate on the simpler semantics.

Definition: 2 *A relation R over $\mathcal{P}$ is said to be a bisimulation if $R(S_1, S_2)$ implies*

$S_1 \xrightarrow{\mu} S_1'$ *implies* $S_2 \xrightarrow{\mu} S_2'$ *and* $R(S_1', S_2')$ *and*

$S_2 \xrightarrow{\mu} S_2'$ *implies* $S_1 \xrightarrow{\mu} S_1'$ *and* $R(S_1', S_2')$

Action Prefix	$\mu\cdot \mathrm{P} \stackrel{\{\mu\}}{\rightarrowtail} \mathrm{P}$
Replication	$\dfrac{\mathrm{P} \stackrel{O_p}{\rightarrowtail} \mathrm{P}',\ \mathrm{Q} \stackrel{O_q}{\rightarrowtail} \mathrm{Q}' \quad O = O_p + O_q}{(\mathrm{P} \amalg \mathrm{Q}) \stackrel{O}{\rightarrowtail} (\mathrm{P}' \amalg \mathrm{Q}')}$
Non-Determinism	$\dfrac{\mathrm{P} \stackrel{O}{\rightarrowtail} \mathrm{P}'}{(\mathrm{P} + \mathrm{Q}) \stackrel{O}{\rightarrowtail} \mathrm{P}' \quad (\mathrm{Q} + \mathrm{P}) \stackrel{O}{\rightarrowtail} \mathrm{P}'}$
Parallel Composition	$\dfrac{\mathrm{P} \stackrel{O}{\rightarrowtail} \mathrm{P}'}{(\mathrm{P} \mid \mathrm{Q}) \stackrel{O}{\rightarrowtail} (\mathrm{P}' \mid \mathrm{Q}) \quad (\mathrm{Q} \mid \mathrm{P}) \stackrel{O}{\rightarrowtail} (\mathrm{Q} \mid \mathrm{P}')}$

Figure 1: Internal Moves

Seal	$\dfrac{\mathrm{P} \stackrel{O}{\rightarrowtail} \mathrm{P}',\ \mu \in \mathsf{Voted\_Action}\ (O)}{(\mathrm{P} \circledcirc) \stackrel{\mu}{\longrightarrow} (\mathrm{P}' \circledcirc)}$
Action Prefix	$\mu\cdot \mathrm{S} \stackrel{\mu}{\longrightarrow} \mathrm{S}$
Non-Determinism	$\dfrac{\mathrm{S}_1 \stackrel{\mu}{\longrightarrow} \mathrm{S}'}{(\mathrm{S}_1 + \mathrm{S}_2) \stackrel{\mu}{\longrightarrow} \mathrm{S}' \quad (\mathrm{S}_2 + \mathrm{S}_1) \stackrel{\mu}{\longrightarrow} \mathrm{S}'}$
Parallel Composition	$\dfrac{\mathrm{S}_1 \stackrel{\mu}{\longrightarrow} \mathrm{S}'}{(\mathrm{S}_1 \mid \mathrm{S}_2) \stackrel{\mu}{\longrightarrow} (\mathrm{S}' \mid \mathrm{S}_2) \quad (\mathrm{S}_2 \mid \mathrm{S}_1) \stackrel{\mu}{\longrightarrow} (\mathrm{S}_2 \mid \mathrm{S}')}$

Figure 2: External Moves

Definition: 3 *Let* $\sim = \bigcup \{R \mid R \text{ is a bisimulation}\}$

Proposition 1 *The usual laws about* $\sim$ *hold, i.e.,* $\sim$ *is the largest bisimulation relation, it is an equivalence, it is a congruence,* $(S + \mathbf{0}) \sim (S \mid \mathbf{0}) \sim S$*,* $+, \mid$ *are commutative, associative with respect to* $\sim$*.*

In providing a sound and complete axiomatisation of the bisimulation equivalence, we need to consider two sets of equations; one for elements of $\mathcal{P}_R$ and the other for elements of $\mathcal{P}$.

As the voting process is synchronous, the axiomatisation is simplified if one extends the syntax of $\mathcal{P}_R$ to include non-trivial(non-empty) multi-set prefixes, i.e., replace $(\mu\cdot \mathrm{P})$ by $(m\cdot \mathrm{P})$ where m is a non empty multi-set. The internal operational rule for action prefix is replaced to specify multi-set prefix and is $(m\cdot \mathrm{P}) \stackrel{m}{\rightarrowtail} \mathrm{P}$.

The proof rules for bisimulation are given in figures 3 and 4.

The set of rules $(=_p)$ in figure 3 identifies terms over $\mathcal{P}_R$. The set of rules $(=)$ in figure 4 identifies terms over $\mathcal{P}$ and uses $=_p$.

The proof that the above set of rules completely characterise the bisimulation equivalence is standard. The reader is referred to [18] for the details of the proof technique. We define two standard forms *replicated* standard form and *standard* form for elements in $\mathcal{P}_R$ and $\mathcal{P}$ respectively.

Definition: 4 $\mathbf{0}$ *is is replicated standard form.* $\sum_i m_i \cdot P_i$ *is in replicated standard form if each* P_i *is in replicated standard form.*

$(\mathbf{0} \circledcirc)$ *is in standard form.* $\sum_i \mu_i \cdot S_i$ *is in standard form if each* S_i *is in standard form.*

Proposition 2 *Every process in* $\mathcal{P}$ *can be converted using the given rules to an equivalent process which is in standard form.*

Proof: As the proof technique is standard we only show that $((\mathrm{P} \amalg \mathrm{Q})\circledcirc)$ where P and Q are in replicated standard form can converted to standard form.

As P and Q are in replicated standard from, we can convert $(\mathrm{P} \amalg \mathrm{Q})$ to the form $\sum_i m_i \cdot R_i$ using the

Identity $(\mathrm{P} + \mathbf{0}) =_p \ (\mathrm{P} \mid \mathbf{0}) =_p \ \mathrm{P}$

Idempotence $(\mathrm{P} + \mathrm{P}) =_p \ \mathrm{P}$

Commutativity $(\mathrm{P} + \mathrm{Q}) =_p \ (\mathrm{Q} + \mathrm{P})$, $(\mathrm{P} \mid \mathrm{Q}) =_p (\mathrm{Q} \mid \mathrm{P})$

Associativity $(\mathrm{P} + \mathrm{Q}) + \mathrm{R} =_p \ \mathrm{P} + (\mathrm{Q} + \mathrm{R})$ and $(\mathrm{P} \mid \mathrm{Q}) \mid \mathrm{R} =_p \ \mathrm{P} \mid (\mathrm{Q} \mid \mathrm{R})$

Replication $(\sum_i m_i \cdot P_i) \amalg (\sum_j n_j \cdot Q_j) =_p \ \sum_{i,j} (m_i + n_j) \cdot (P_i \amalg Q_j)$

Interleaving Let P be $\sum_i m_i \cdot P_i$ and Q be $\sum_j n_j \cdot Q_j$.

$$(\mathrm{P} \mid \mathrm{Q}) =_p \ \sum_i m_i \cdot (P_i \mid Q) + \sum_j n_j \cdot (P \mid Q_j)$$

Figure 3: Rules for terms over $\mathcal{P}_R$

Seal $\mathrm{P} =_p \ \mathrm{Q}$ implies $(\mathrm{P} \circledcirc) = (\mathrm{Q} \circledcirc)$

Idempotence $(\mathrm{P} \amalg \mathrm{P})\circledcirc = (\mathrm{P} \circledcirc)$

Distributivity $(\mathrm{P} + \mathrm{Q})\circledcirc = (\mathrm{P}\circledcirc + \mathrm{Q}\circledcirc)$

Vote $(m \cdot \mathrm{P}) \circledcirc = \sum_{\mu \in \mathsf{Voted\_Action}(m)} \mu \cdot (\mathrm{P} \circledcirc)$

Identity $(\mathrm{S} + \mathbf{0}) = (\mathrm{S} \mid \mathbf{0}) = \mathrm{S}$

Idempotence $(\mathrm{S} + \mathrm{S}) = \mathrm{S}$

Commutativity $(\mathrm{S}_1 + \mathrm{S}_2) = (\mathrm{S}_2 + \mathrm{S}_1)$ and $(\mathrm{S}_1 \mid \mathrm{S}_2) = (\mathrm{S}_2 \mid \mathrm{S}_1)$

Associativity $(\mathrm{S}_1 + \mathrm{S}_2) + \mathrm{S}_3 = \mathrm{S}_1 + (\mathrm{S}_2 + \mathrm{S}_3)$ and $(\mathrm{S}_1 \mid \mathrm{S}_2) \mid \mathrm{S}_3 = \mathrm{S}_1 \mid (\mathrm{S}_2 \mid \mathrm{S}_3)$

Interleaving Let S be $\sum_i \mu_i \cdot S_i$ and T be $\sum_j \nu_j \cdot T_j$.

$$(\mathrm{S} \mid \mathrm{T}) = \sum_i \mu_i \cdot (S_i \mid T) + \sum_j \nu_j \cdot (S \mid T_j)$$

Figure 4: Rules for terms over $\mathcal{P}$

Non-Empty Prefix	$\frac{\mu \in \text{Voted\_Action}\ (\xi)}{\xi \cdot \mathrm{P} \xrightarrow{\mu} \mathrm{P}}$
Empty-Prefix	$\frac{\xi = \emptyset,\ \mathrm{P} \xrightarrow{\mu} \mathrm{P}'}{\xi \cdot \mathrm{P} \xrightarrow{\mu} \mathrm{P}'}$

Figure 5: New Semantics

Replication rule in figure 3. By using the **Vote** rule in figure 4 we can convert $(\sum_i m_i \cdot R_i \circledcirc)$ to standard form. □

Proposition 3 *The set of equations* $=_p$ *and* $=$ *completely axiomatise the* $\sim$ *relation.*

Proof: Using proposition 2, we can covert every process to a standard form. So for completeness we need to consider only standard forms. If P and Q are in standard form and $\mathrm{P} \sim \mathrm{Q}$, then using the idempotence, commutativity and associativity of '+', we can show that $\mathrm{P} = \mathrm{P} + \mathrm{Q} = \mathrm{Q} + \mathrm{P} = \mathrm{Q}$. □

The theory developed so far has been a simple extension to a subset of CCS. In the remainder of the paper we develop a theory which is directly relevant to fault-tolerant systems.

3 Fault Preorders

In the above section we have presented a syntax and semantics for replicated processes. The external behaviour of such a system was similar to that of CCS (i.e., the replication was transparent). This can be interpreted to be a user's view point where fault tolerant aspects such as replication are hidden. This being satisfactory for a user, the above semantics is not directly relevant to the designer of robust systems. For a theory to be useful in the design and analysis of fault tolerant systems, the effect of fault introduction in a system and the effect of introduced faults on observable behaviour needs to be developed. While a completely fault tolerant system is desired, it is possible that a system may fail. The operating environment may cause more faults that the system was designed to overcome. It is still necessary to study the behaviour of such failed systems and compare them against the intended behaviour. The study of failed systems along with a notion of fault injection can be used to study fault tolerance. If a system S_1 is more fault tolerant than a system S_2 within a given fault model, the system S_1 injected with faults will be less faulty than system S_2 injected with identical faults.

In the remainder of the paper we develop a framework in which the effect of already introduced faults on observable behaviour can be studied.

To characterise the effect of faults on systems, we consider the following simplified syntax

$$\mathrm{P} ::= \mathbf{0} \mid \xi \cdot \mathrm{P} \mid (\mathrm{P} + \mathrm{P})$$

where ξ is a multiset (possibly empty) over Λ.

The above syntax is almost identical to the *replicated* standard form defined for elements of $\mathcal{P}_R$. We permit empty multisets as the non-empty multisets can be reduced by faults to the empty set. For example, a single omission fault will alter $\{\mu\} \cdot \mathrm{P}$ to $\emptyset \cdot \mathrm{P}$. It was not necessary to consider empty prefixes in the replicated standard form as the multi-set prefixes were obtained from action prefixes and were guaranteed to be non-empty. It is necessary to consider elements of $\mathcal{P}_R$ as opposed to elements of $\mathcal{P}$ as the degree of replication is important. We consider elements in *replicated* standard form to simplify the exposition.

We no longer have two semantic relations and the operational semantics (indicated by $\longrightarrow$) for $(\xi \cdot \mathrm{P})$ is presented in figure 5. The rules for + are identical to those given in figure 2.

If the multiset prefix is non-empty it exhibits the action that has received maximum number of votes, while if the multi-set prefix is empty, it is effectively discarded.

We are interested in "good" environments, i.e., environments where all votes are identical. In such an environment, all the voting sub-systems reach a consensus on the action to be exhibited. Similarly we define a perfect process where consensus is reached for every behaviour.

Definition: 5 *A multiset* ξ *is said to be perfect iff* $\exists\ \mu_1 \in \Lambda$ *such that* $\xi(\mu_1) > 0$ *and* $\forall\ \mu_2 \in (\Lambda - \{\mu_1\})$, $\xi(\mu_2) = 0$.

A process P is perfect if all multisets that occur in it are perfect.

Based on the observational operational relation we use the following abbreviations.

Definition: 6 *$P \xrightarrow{\mu}$ iff $\exists\ P'$ such that $P \xrightarrow{\mu} P'$ and $P \not\longrightarrow$ otherwise.*

Given that replicated systems can be defined, we now describe fault introduction. In the framework we develop, the result of introducing a fault to a replicated process is another replicated process. The resulting process represents the behaviour of the faulty system, i.e., after it has been affected by a fault. The modification of a process depends on the type of fault one wishes to model. For example, if the correct system is $(\mu^3 \cdot \mathrm{P})$, a single μ-omission fault will transform it to $(\mu^2 \cdot \mathrm{P})$ while a μ-μ' garbling fault will transform the given process to $(\{\mu^2, \mu'\} \cdot \mathrm{P})$. The idea of fault introduction is similar to that of refinement [19, 2]. However, they place restrictions on the behaviour of the refinement operators. Hence the results presented are not directly applicable. We also represent faults as a refinement function. But their application to processes is different. The exact definition of fault introduction will be presented later. In general, fault introduction can be defined as follows.

Definition: 7 *Let ϱ be a fault refinement and P be a process. Define $(P \dagger \varrho)$ as follows:*

$\mathbf{0} \dagger \varrho = \mathbf{0}$, $(\xi \cdot P) \dagger \varrho = \varrho(\xi) \cdot P$, $(P + Q) \dagger \varrho = (P \dagger \varrho) + (Q \dagger \varrho)$

Intuitively, if a process has terminated, no fault can affect it, while if a process can perform an action, an occurrence of a fault could alter the action. The presence of non-determinism does not reduce the effect of the fault. The exact definition $\varrho(\xi)$ will depend on the nature of ϱ and will be discussed later.

The above definition of fault introduction, affects only the first action a process can exhibit and hence models the occurrence of a single transient failure. This is in keeping with the philosophy of modelling faults as special operations [5].

The idea of approximations as a frame work for verifying satisfaction of specifications by implementations is well known. These approximations can be in the form of a preorder where $(\mathrm{P} \sqsubseteq \mathrm{Q})$ means that any move P makes can be matched by Q. Therefore, if P is an implementation and Q is a specification, $(\mathrm{P} \sqsubseteq \mathrm{Q})$ requires that all behaviours of an implementation are valid given the specification.

Observational preorders (like trace and testing [9]) have been defined for process calculi. In general, for processes P and Q, $(\mathrm{P} \sqsubseteq \mathrm{Q})$ implies that every behaviour of P can be matched by Q. For example, let P be $(\mu_1 \cdot \mu_2 \cdot \mu_3 \cdot \mathbf{0} + \mu_1 \cdot \mu_2 \cdot \mu_4 \cdot \mathbf{0})$ and Q be $\mu_1 \cdot (\mu_2 \cdot \mu_3 \cdot \mathbf{0} + \mu_2 \cdot \mu_4 \cdot \mathbf{0})$. P is less than Q in the trace preorder as the traces of P is included in the traces of Q. Similarly, P is less than Q in the testing preorder as every test (i.e., reacting to external stimuli [9]) that P passes, Q can also pass. Depending on the notion of behaviour different preorders can be obtained.

Both the trace and testing preorders are based on the observable behaviour of a process. However such preorders are not directly useful in the fault-tolerant setting. If $\mathrm{P} \prec \mathrm{Q}$ is to mean that Q can withstand at-least as many faults as P, then the processes with the faults cannot be related based only on observations. If P is μ and Q is μ^3, under a value-altering fault of μ to μ_1 P can exhibit μ_1 which Q cannot match. Therefore, P affected by a fault is not observationally related to Q affected by a fault.

As the behaviour of a faulty process can be significantly different from its behaviour in the absence of faults, a 'correctness' condition is necessary. The correctness criterion distinguishes faulty behaviour from non-faulty behaviour. When relating two processes only the correct behaviour needs to be matched. This indicates the need for an indexed relation. [16] introduces the idea of equivalences induced by contexts called relativised bisimulation. For example, $\mathrm{P} \prec_C \mathrm{Q}$ relates the behaviours of P and Q in the context C. We use this idea with a different interpretation in developing the fault preorders. The preorders we consider do not directly deal with fault-tolerance. They characterise faulty systems, i.e., where faults are already introduced.

In the next few sections we develop the various fault preorders. Each type of fault induces a different preorder. This is natural, as the behaviour of a system with omission failures will not be identical to a system with addition failures. In this work we consider omission faults, value garbling faults and addition faults. We define indexed preorders of the form $\mathrm{P} \prec_C \mathrm{Q}$ where C represents the "correct non-faulty" behaviour. The intuitive interpretation is that if C can make a move which P or Q cannot match, one can assume that it is due to the occurrence of a fault. In the context of omission faults one can assume that P or Q has jumped ahead. while in the context of addition faults, one can assume that P or Q needs to be stepped to reach the same state as C.

In the next three section we develop the preorders for three types of faults.

3.1 Omission Faults

An omission fault in communicating systems is characterised by a unit not sending a message it had to. In our context, an omission fault is represented by a process not exhibiting a required action.

Definition 8 defines the preorder induced by omission failures.

Definition: 8 *$P \prec^{O}_{C} Q$ iff $C \xrightarrow{\mu} C'$ then*

If $P \xrightarrow{\mu} P'$ *then* $\exists\ Q'$ *such that* $(Q \xrightarrow{\mu} Q')$ *and* $(P' \prec^O_{C'} Q')$

If $P \not\xrightarrow{\mu}$ *and* $Q \xrightarrow{\mu}$, *then* $\exists\ Q'$ *such that* $(Q \xrightarrow{\mu} Q')$ *and* $(P \prec^O_{C'} Q')$

If $P \not\xrightarrow{\mu}$ *and* $Q \not\xrightarrow{\mu}$, *then* $P \prec^O_{C'} Q$

We use C as the driving agent. If C an perform an action and P can match it, then Q must be able to match the move. This ensures that if P can behave correctly then so can Q. If P cannot match the move (due to omission failure) then Q may (no fault) or may not (omission fault) be able to match it. If P (and/or Q) cannot match the move, they are 'held stationary' and C exhibits an action; thus formalising the intuition behind omission faults.

Example 1 *Let* $P = \mu_1 \cdot \mu_2 \cdot \mu_3 \cdot \mathbf{0}$, $Q = \mu_1 \cdot \mu_4 \cdot \mu_2 \cdot \mu_3 \cdot \mathbf{0}$ *and* $C = \mu_1 \cdot \mu_4 \cdot \mu_2 \cdot \mu_3 \cdot \mu_5 \cdot \mathbf{0}$. *As P has lost* μ_4 *and* μ_5 *and Q only* μ_5, $(P \prec^O_C Q)$.

Let $P = \mu_1 \cdot \mu_1 \cdot \mathbf{0}$, $Q = \mu_1 \cdot \mu_2 \cdot \mu_1 \cdot \mathbf{0}$ *and* $C = \mu_1 \cdot (\mu_2 \cdot \mu_1 \cdot \mathbf{0} + \mu_1 \cdot \mathbf{0})$. *P can be derived from C by omitting* μ_2 *in the first option and* μ_1 *in the second option while Q can be derived from C by omitting* μ_1 *in the second option. This would seem to indicate that* $(P \prec^O_C Q)$. *However, this is not the case as P can simulate the C's second option while Q cannot. This indicates that in the presence of non-determinism omission faults can result in correct options. If Q were* $\mu_1 \cdot (\mu_2 \cdot \mathbf{0} + \mu_1 \cdot \mathbf{0})$ *then* $(P \prec^O_C Q)$.

Proposition 4 $\prec^O_C$ *is a preorder i.e., is reflexive and transitive,* $(P \prec^O_{\mathbf{0}} Q)$ *and* $(\mathbf{0} \prec^O_C Q)$

Proof: (P $\prec^O_C$ P) is obvious as it will use only the first and third clauses of the definition. To prove the transitivity of $\prec^O_C$ is straightforward. As **0** has no move, P $\prec^O_{\mathbf{0}}$ Q is direct.

0 $\prec^O_C$ Q requires some comment. As $\mathbf{0} \not\longrightarrow$, only the second and third clause of the definition are relevant. If for C $\xrightarrow{\mu}$ C′ there exists Q′ such that Q $\xrightarrow{\mu}$ Q′ then by an inductive argument we can show that $\mathbf{0} \prec^O_{C'}$ Q′. Otherwise we can show that **0** $\prec^O_{C'}$ Q. As C is finite, we will eventually arrive at **0** $\prec^O_{\mathbf{0}}$ Q which is true. □

The **0** process can be perceived as the result of a process which has been erased by a large number of omission faults and hence is a least element in the preorder. (P $\prec^O_{\mathbf{0}}$ Q) is valid as the pre-order is indexed by the correctness condition and we constrain the behaviour to a pattern dictated by it. As **0** can exhibit no action, any two processes are related by $\prec^O_{\mathbf{0}}$. In general, if (P $\prec^O_C$ Q), both P and Q could have unrelated extraneous behaviours as shown by the following example.

Example 2 *Let C be* $\mu_1 \cdot \mathbf{0}$, *P be* $(\mu_2 \cdot \mathbf{0} \mid \mu_3 \cdot \mathbf{0})$ *and Q be* $(\mu_1 \cdot \mathbf{0} \mid \mu_4 \cdot \mathbf{0})$. $(P \prec^O_C Q)$ *as for the* μ_1 *move of C, P could exhibit* μ_2 *and Q could exhibit* μ_1. *As C evolves to* **0**, $(\mu_3 \cdot \mathbf{0}) \prec^O_{\mathbf{0}} (\mu_4 \cdot \mathbf{0})$. *If the above relation is not desired, a generalisation based on modal transition systems [17] can be used. This needs further work. and will be reported elsewhere.*

The definition of P $\prec^O_C$ Q assumes that faults have been introduced into P and Q and does not require that P is no more *fault-tolerant* than Q. It only indicates that Q is no more *faulty* than P. To define fault-tolerance, we need to define fault introduction and hence need to define $\varrho(\xi)$.

We consider ϱ to be an μ_1-omission fault, if it erases μ_1, i.e., $\varrho(\mu_1) = \mathbf{0}$ without affecting any other action.

Definition: 9 *Let* ξ *be a multiset and* ϱ *be an* μ_1 *omission fault.* $\varrho(\xi) = \xi'$ *where*

$\xi'(\mu) = \xi(\mu)$ *if* $\mu \neq \mu_1$ *and*

$\xi'(\mu_1) = (\xi(\mu_1) \dot{-} 1)$ *where* $\dot{-}$ *is the monus operation.*

Only the votes obtained by the action μ_1 is affected by an μ_1-omission fault introduction. The fault-introduction reduces by one the number of votes received by μ_1.

Theorem 1 *Assume that* ξ_p *and* ξ_q *are perfect and* ϱ *an* μ *omission refinement.*

If $\xi_p \cdot P \xrightarrow{\mu} P$, *and* $(\ (\xi_p \cdot P) \dagger \varrho) \prec^O_{\xi_p \cdot P} (\ (\xi_q \cdot Q) \dagger \varrho)$, *then* $\xi_q(\mu) \geq 2$ *or* $\xi_p(\mu) = \xi_q(\mu)$.

Proof: As ξ_p and ξ_q are perfect and $(\xi_p \cdot$P) exhibits μ, $\xi_p(\mu)$ is greater than zero. From definition 7, $(\xi_p \cdot$P) $\dagger\varrho = \varrho(\xi_p) \cdot$P and $(\xi_q \cdot$Q) $\dagger\varrho = \varrho(\xi_q) \cdot$Q. By the definition of omission refinement, ϱ reduces the vote of μ.

If $(\ (\xi_p \cdot$P) $\dagger\varrho)$ can exhibit μ, so can $(\ (\xi_q \cdot$Q) $\dagger\varrho)$. Hence $\varrho(\xi_q)(\mu) \geq 1$ and hence $\xi_q(\mu) \geq 2$.

If $(\ (\xi_p \cdot$P) $\dagger\varrho)$ cannot exhibit μ, the following two cases arise. If the process $(\ (\xi_q \cdot$Q) $\dagger\varrho)$ can exhibit μ, by above argument $\xi_q(\mu) \geq 2$. If the process $(\ (\xi_q \cdot$Q) $\dagger\varrho)$ cannot exhibit μ, then $\xi_p(\mu) = \xi_q(\mu) = 1$. □

The above theorem reiterates the fact that a single replication (or two units) is sufficient to withstand a single instantaneous omission fault. The reason we consider only perfect votes is that if we consider a somewhat erroneous system, an omission fault could manifest itself as other faults as illustrated in the following example.

Example 3 *The process $\langle \mu_1^3, \mu_2^2 \rangle$ with a single μ_1 omission fault will behave as $\langle \mu_1^2, \mu_2^2 \rangle$. Hence it can exhibit μ_2 which will then be confused with a μ_1 to μ_2 value fault.*

3.2 Value Faults

Definition 10 defines the preorder induced by value (also called garbling) faults; i.e., faults which alter a correct action μ to an incorrect action μ_1.

Definition: 10 *$P \prec_C^V Q$ iff $C \xrightarrow{\mu} C'$ then*

If $P \xrightarrow{\mu} P'$ then $\exists\ Q'$ such that $(Q \xrightarrow{\mu} Q')$ and $(P' \prec_{C'}^V Q')$

If $P \not\xrightarrow{\mu}$ and $Q \xrightarrow{\mu}$ then $\exists\ \mu_1, P', Q'$ such that $(P \xrightarrow{\mu_1} P')$ and $(Q \xrightarrow{\mu} Q')$ and $(P' \prec_{C'}^V Q')$

If $P \not\xrightarrow{\mu}$ and $Q \not\xrightarrow{\mu}$ then $\exists\ \mu_1\ P', Q'$ such that $(P \xrightarrow{\mu_1} P')$ and $(Q \xrightarrow{\mu_1} Q')$ and $(P' \prec_{C'}^V Q')$

The main difference between definition 8 and definition 10 is that if a matching action cannot be exhibited, a different action *needs* to be exhibited. Furthermore, if P $\prec_C^V$ Q and if both P and Q are faulty (i.e., cannot exhibit the correct action), they are required to exhibit *identical* faulty actions, i.e., have identical fault behaviour.

Example 4 *It is easy to see that $\mu_1 \cdot \mu_2 \cdot \mu_4 \cdot \mathbf{0} \prec^V_{\mu_1 \cdot \mu_3 \mu_5 \cdot \mathbf{0}} \mu_1 \cdot \mu_3 \cdot \mu_4 \cdot \mathbf{0}$. The lesser process has suffered two value faults while the better process has suffered only one fault.*

Note that it is possible to 'forget' non-determinism. For example, $\mu_1 \cdot \mu_2 \cdot \mu_4 \cdot \mathbf{0}$ (say P) can be derived from $\mu_1 \cdot (\mu_2 \cdot \mu_3 \cdot \mathbf{0} + \mu_3 \cdot \mu_4 \cdot \mathbf{0})$ (say C) from purely value faults. The fault alters μ_3 to μ_4 in the first option and μ_3 to μ_2 in the second option. Though, $\mu_1 \cdot \mu_2 \cdot \mu_3 \cdot \mathbf{0}$ (say Q) appears to be less faulty than P (only the second option of C is altered to $\mu_2 \cdot \mu_3 \cdot \mathbf{0}$ while the first option is left untouched) it is not the case that $P \prec_C^V Q$. This is because if C chose the second option, then P does not suffer from μ_4 to μ_3 fault which Q suffers from. As in example 1 the presence of non-determinism can obscure the intuitive notion of faulty processes.

Proposition 5 *$\prec_C^V$ is a preorder and $(P \prec_{\mathbf{0}}^V Q)$*

Note that it is not the case ($\mathbf{0} \prec_C^V$ Q) in general, as if C $\xrightarrow{\mu}$ and Q $\not\xrightarrow{\mu}$, both $\mathbf{0}$ and Q are required to make a move. This is because we have indexed the preorder by C. Alternatively a definition using P as the index or by adding ($\mathbf{0} \prec_C^V$) to it can be considered. The advantages of such a definition needs further investigation.

As in the omission case, we have to define fault injection, for which $\varrho(\xi)$ has to be defined. We consider a refinement function ϱ to be a μ_1-μ_2 value fault, if it alters μ_1 to μ_2, i.e., $\varrho(\mu_1) = \mu_2$, while having no effect on other actions.

The introduction of a value fault to a system is defined below.

Definition: 11 *Let ξ be a multi-set and ϱ a μ_1-μ_2 value fault. Define $\varrho(\xi) = \xi'$ where*

$$\xi'(\mu) = \begin{cases} \xi(\mu_1) \dot{-} 1 & \mu = \mu_1 \\ \xi(\mu_2) & \mu = \mu_2 \text{ and } \xi(\mu_1) = 0 \\ \xi(\mu_2) + 1 & \mu = \mu_2 \text{ and } \xi(\mu_1) > 0 \\ \xi(\mu) & \text{otherwise} \end{cases}$$

The above definition considers a μ_1-μ_2 fault and increments the vote of μ_2 only if μ_1 received a positive vote. If μ_1 received no votes, it is not possible to garble it.

Theorem 2 *Let $\xi_p \cdot P \xrightarrow{\mu_1} P$ and $\xi_q \cdot Q \xrightarrow{\mu_1} Q$. Also, let $\forall\ \mu \in \Lambda$- $\{\mu_1, \mu_2\}$, $\xi_p(\mu) = \xi_q(\mu) = 0$.*
Let ϱ be a μ_1-μ_2 value fault.
If $(\ (\xi_p \cdot P) \dagger \varrho) \prec^V_{\xi_p \cdot P} (\ (\xi_q \cdot Q) \dagger \varrho)$, then $(\xi_p(\mu_1) - \xi_p(\mu_2) \leq 1)$ or $(\xi_q(\mu_1) - \xi_q(\mu_2) > 1)$

Proof: If $(\ (\xi_p \cdot \mathrm{P}) \dagger \varrho)$ has no μ_1 move then $0 \leq \xi_p(\mu_1) - \xi_p(\mu_2) \leq 1$. This is because if the difference were larger than 1, subtracting one form μ_1 and adding one to μ_2 cannot prevent the exhibition of μ_1. Also, as an μ_1 move was possible from $(\xi_p \cdot \mathrm{P})$, it cannot be negative.

If $(\ (\xi_p \cdot \mathrm{P}) \dagger \varrho)$ has an μ_1 move then so does $(\ (\xi_q \cdot \mathrm{Q}) \dagger \varrho)$. Hence, adding one vote to μ_2 and subtracting one vote from μ_1 does not prevent the exhibition of μ_1. Hence $\leq \xi_q(\mu_1) - \xi_q(\mu_2) > 1$. □

The above theorem indicates if $(\xi_q \cdot \mathrm{Q})$ is at least as fault-tolerant as $(\xi_p \cdot \mathrm{P})$, the difference in votes for μ_1 and μ_2 in no less than P's difference in votes.

The converse of the above theorem is also true. The converse theorem will not hold if ξ_p or ξ_q had 'significant' votes for other actions as illustrated by the following example.

Example 5 *The process* $\langle \mu_1^3, \mu_2^1, \mu_3^2 \rangle$ *under* ϱ *can exhibit* μ_1*, while* $\langle \mu_1^4, \mu_3^4 \rangle$ *under* ϱ *cannot. Though the value fault changed* μ_1 *to* μ_2 *the presence of* μ_3 *changes the nature of the fault.*

3.3 Addition Faults

The treatment of addition faults is different from the treatment of omission and value faults. In communicating systems, an addition fault adds a message to the system. As we are considering a frame-work with votes, the issue of *when* the additional message arrives is crucial. It is possible to assume no bound on when the additional message could arrive. Under such an assumption, the theory becomes unwieldy. In this paper we consider an 'atomic' semantics, i.e., assume that the additional message arrives along with the actual message.

Intuitively, ϱ models an μ addition fault if it increments the number of votes received by μ by one while not affecting the other actions. We define the effect of an addition fault on the current state of votes as follows.

Definition: 12 *A refinement* ϱ *is an* μ *addition fault if*

$$\forall\ \xi,\ \varrho(\xi) = \xi' \text{ where } \xi'(\mu) = \xi(\mu) + 1 \text{ and } \forall\ \mu_1 \neq \mu,\ \xi'(\mu_1) = \xi(\mu_1).$$

The above definition ensures that the 'correct' action and the 'faulty' additional action are considered by the voting mechanism *simultaneously.*

Fault introduction via addition refinement will be observationally similar to value-fault as one action can be altered to another. If a system exhibits μ_1 instead of the expected μ an observer cannot determine if the fault was due to garbling or addition. Hence the definition of the preorder induced by addition-faults is identical to definition 10. Definition 13 describes the preorder induced by addition faults and is presented only for the sake of completeness.

Definition: 13 $P \prec_C^A Q$ *iff* $C \xrightarrow{\mu} C'$ *then*

If $P \xrightarrow{\mu} P'$ *then* $\exists\ Q'$ *such that* $(Q \xrightarrow{\mu} Q')$ *and* $(P' \prec_{C'}^A Q')$

If $P \not\xrightarrow{\mu}$ *and* $Q \xrightarrow{\mu} Q'$ *then* $\exists\ \mu_1,\ P'$ *such that* $(P \xrightarrow{\mu_1} P')$ *and* $(P' \prec_{C'}^A Q')$

If $P \not\xrightarrow{\mu}$ *and* $Q \not\xrightarrow{\mu}$ *then* $\exists\ \mu_1,\ P'$, Q' *such that* $(P \xrightarrow{\mu_1} P')$ *and* $(Q \xrightarrow{\mu_1} Q')$ *and* $(P' \prec_{C'}^A Q')$

Proposition 6 $\prec_C^A$ *is a preorder,* $P \prec_{\mathbf{0}}^A Q$

As in the value fault case, **0** is not the least element in the preorder.

Proposition 7 *Let* $\xi_p \cdot P \xrightarrow{\mu} P'$ *and let* ϱ *be an* μ_1 *addition fault.*
If $(\ (\xi_p \cdot P) \dagger \varrho) \prec_{\xi_p \cdot P}^A (\ (\xi_q \cdot Q) \dagger \varrho)$*, then* $(\xi_p(\mu) - \xi_p(\mu_1) < 1)$ *or* $(\xi_q(\mu) - \xi_q(\mu_1) > 1)$

The above proposition is similar to theorem 2 and the proof is very similar.

This concludes the definition of the fault preorders. In the next section we present a modal logic characterisation of the omission and value fault preorders.

4 Modal Characterisation

It has been shown that the usual bisimulation semantics for process algebras can be characterised by the modal-μ calculus [23]. In this section we characterise certain aspects of fault-tolerance using a subset of the modal-μ logic. The fragment of the modal-μ we use is as follows

$$\varphi ::= \text{True} \mid \langle \mu \rangle \varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

We do not consider negation or the necessity modality, the reason is that we have been unable to meaningfully describe the effects of faults on formulae involving negation. Details of this are presented in section 5.

Associated with the logical formulae and the set of processes is a satisfaction relation. A process P satisfies a formula $\langle \mu \rangle \varphi'$ (written as $P \models \langle \mu \rangle \varphi'$) iff there is a P' such that $P \xrightarrow{\mu} P'$ and $(P' \models \varphi')$. All processes satisfy True while $\wedge$ and $\vee$ represent logical 'conjunction' and logical 'disjunction' respectively.

In the following two sections we show how the omission and value fault preorders can be logically described. We do not consider addition faults as the preorder associated with addition fault is identical to value fault preorder.

4.1 Omission Faults

As an omission fault introduction can prevent a process from exhibiting an initial action, the following proposition holds.

Proposition 8 *Let P be perfect. If* $P \models \langle\mu\rangle\varphi$ *and* ϱ *an* μ*-omission, then* $(P \dagger \varrho) \models \langle\mu\rangle\varphi \vee \varphi$ *and if* ϱ *is an* μ_1 *omission and* $\mu \neq \mu_1$, $(P \dagger \varrho) \models \langle\mu\rangle\varphi$.

Proof: As P is perfect and $P \models \langle\mu\rangle\varphi$ $P \xrightarrow{\mu}$. This implies that P has a subterm of the form $\xi \cdot P_1$ such that $\xi(\mu) \geq 1$ and $P_1 \models \varphi$.

If ϱ reduces the vote of μ by one then either $\varrho(\xi)(\mu) = 0$ or $\varrho(\xi)(\mu) > 0$. In the first case $\varrho(\xi)\cdot P_1$ will satisfy φ while in the second case $\varrho(\xi)\cdot P_1$ will satisfy $\langle\mu\rangle\varphi$.

If ϱ does not alter the vote of μ $\varrho(\xi) = \xi$. □

As we are considering only 'possible' formulae, non-determinism does not affect the above proposition. For example, let P be $(\mu_1 \cdot \mu_2 \cdot \mathbf{0} + \mu_3 \cdot \mathbf{0})$. P will satisfy $\langle\mu_1\rangle\langle\mu_2\rangle$True and and P under an μ_1 omission will satisfy $\langle\mu_2\rangle$True.

The above proposition gives us some insight into the logical structure of the fault preorder. Given a formula we identify formulae which are derived by inserting appropriate faults at all possible points. Towards the formal description of the fault preorder $\prec_C^O$, define a translation function $[\![\varphi]\!]_O$ which identifies the possible formulae that a 'faulty' process can satisfy given that the 'correct' process satisfies φ.

Definition: 14 $[\![True]\!]_O = \{ True\}$

$[\![\langle\mu\rangle\ \varphi]\!]_O = \{\langle\mu\rangle\ \varphi' \mid \varphi' \in [\![\varphi]\!]_O\} \cup [\![\varphi]\!]_O$

$[\![\varphi_1 \oplus \varphi_2\ ]\!]_O = \{\varphi_i \oplus \varphi_j \mid \varphi_i \in [\![\varphi_1]\!]_O$ *and* $\varphi_j \in [\![\varphi_2]\!]_O\}$ *for* $\oplus \in \{\vee, \wedge\}$

The above definition transforms a given formula into ones where omission faults have occurred at arbitrary instances. The translation by itself is not sufficient as if $P \prec_C^O Q$, it need not be the case that all formulae that P satisfies Q will. For instance, let $(C \models \langle\mu_1\rangle\langle\mu_2\rangle$True) and $(P \models \langle\mu_2\rangle$True. If Q is less faulty than P, $(Q \not\models \langle\mu_2\rangle$True) but $(Q \models \langle\mu_1\rangle\langle\mu_2\rangle$True). This indicates the need for a hierarchy of formulae which denotes 'less' faulty. As in the preorder case, the hierarchy has to be indexed by a correctness formula.

Definition: 15 *For every formula* φ, *define an ordering on* $[\![\varphi]\!]_O$ *as follows.*

- $\forall \varphi_1, \varphi_2,\ \varphi_1 \sqsubseteq^O_{True} \varphi_2$
- $\forall\ \varphi' \in [\![\langle\mu\rangle\varphi]\!]_O,\ \varphi' \sqsubseteq^O_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi$
- $\forall\ \varphi_1, \varphi_2 \in [\![\varphi]\!]_O$ *such that* $\varphi_1 \sqsubseteq^O_\varphi \varphi_2$ *implies*
  - $\varphi_1 \sqsubseteq^O_{\langle\mu\rangle\varphi}\ \varphi_2$
  - $\forall\ \mu \in \Lambda,\ \varphi_1 \sqsubseteq^O_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi_2$
  - $\forall\ \mu \in \Lambda,\ \langle\mu\rangle\varphi_1 \sqsubseteq^O_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi_2$

As $\sqsubseteq^O_\varphi$ indicates a fault hierarchy and as all processes satisfy True, any two formulae are related if the correctness criteria is True. Every formula in $[\![\varphi]\!]$ represents a formula that a potentially faulty process could satisfy. Hence all elements of $[\![\varphi]\!]$ are less than φ. The third aspect of the definition deals with 'future' faults. If future behaviour indicates that a formula φ_1 is more faulty than another φ_2 under φ, then both could suffer omission faults and hence are related under $\langle\mu\rangle\varphi$ or the formula $\langle\mu\rangle\varphi_2$ is less faulty than both φ_1 and $\langle\mu\rangle\varphi_1$.

Proposition 9 $\sqsubseteq^O_\varphi$ *is a preorder.*

Proof: To show that $\sqsubseteq^O_\varphi$ is reflexive and transitive. Note that True is always an element of $[\![\ \varphi\ ]\!]_O$. Reflexivity, i.e., $\varphi_1 \sqsubseteq^O_\varphi \varphi_1$, is proved by induction on the size of φ. If φ is True or if φ_1 is True then done. If φ is of the form $\langle\mu\rangle\varphi'$ and if φ_1 is of the form $\langle\mu_1\rangle\varphi_1'$ then two cases arise. If μ is identical to μ_1 then by induction hypothesis $\varphi_1' \sqsubseteq^O_{\varphi'} \varphi_1'$. Otherwise $\varphi_1 \sqsubseteq^O_{\varphi'} \varphi_1$. This is true as φ is finite and eventually will reduce to True. The remainder of the proof follows a similar argument. □

The exact relationship between the ordering of processes and the ordering of logical formulae is described after the preorders for logical formulae for the value fault is developed.

4.2 Value Fault

A garbling fault can alter the initial action that a process can perform. Unlike an omission fault, a value fault ensures that the modified process can exhibit an action if the original process could.

Proposition 10 *Let P be perfect. If* $P \models \langle\mu\rangle\varphi$ *and* ϱ *an* μ-μ_1 *value fault, then* $(P \dagger \varrho) \models \langle\mu\rangle\varphi \vee \langle\mu_1\rangle\varphi$ *and if* ϱ *is an* μ_1-μ_2 *value fault and* $\mu \neq \mu_1$, $(P \dagger \varrho) \models \langle\mu\rangle\varphi$.

Proof: The proof is similar to proposition 8. □

The above proposition characterises value-fault introduction. As for the omission faults case we define sets of formulae equipped with an ordering which characterises the fault preorder $\prec_C^V$.

Definition: 16 $[\![True]\!]_V = \{ True \}$

$[\![\langle\mu\rangle\ \varphi]\!]_V = \{\langle\mu_1\rangle\ \varphi' \mid \varphi' \in [\![\varphi]\!]_V \text{ for all } \mu_1 \in \Lambda\}$

$[\![\varphi_1 \oplus \varphi_2\ ]\!]_V = \{\varphi_i \oplus \varphi_j \mid \varphi_i \in [\![\varphi_1]\!]_V \text{ and } \varphi_j \in [\![\varphi_2]\!]_V \text{ for } \oplus \in \{\vee, \wedge\}\}$

Definition: 17 *For every formula* φ, *define an ordering on* $[\![\varphi]\!]_V$ *as follows.*

- $\forall \varphi_1, \varphi_2,\ \varphi_1 \sqsubseteq^V_{True}\ \varphi_2$
- $\forall\ \varphi' \in [\![\langle\mu\rangle\varphi]\!]_V\ \varphi' \sqsubseteq^V_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi$.
- $\forall \varphi_1, \varphi_2 \in [\![\varphi]\!]_V$, *such that* $\varphi_1 \sqsubseteq^V_\varphi\ \varphi_2$, *define*
  - $\forall\ \mu_1 \in \Lambda,\ \langle\mu_1\rangle\varphi_1 \sqsubseteq^V_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi_2$
  - $\forall\ \mu_1 \in \Lambda,\ \langle\mu_1\rangle\varphi_1 \sqsubseteq^V_{\langle\mu\rangle\varphi}\ \langle\mu_1\rangle\varphi_2$

The definition of $\sqsubseteq^V_\varphi$ is similar to that of $\sqsubseteq^O_\varphi$ except in the third case where instead of action omission, we alter μ to μ_1 in φ_1, indicating a garbling fault. In the third clause, φ_2 represents potential garbling in the future.

Proposition 11 $\sqsubseteq^V_\varphi$ *is a preorder.*

The following proposition indicates that the modal formulae equipped with the appropriate ordering captures the fault hierarchy. As Q satisfies a formula 'higher up' in the preorder, Q is less faulty than P.

Theorem 3 *Let* $P \prec^X_C Q$. *and* $C \models\varphi$, *for* $X \in \{O, V\}$. $\forall\varphi' \in [\![\varphi]\!]_X\ P \models\varphi'$ *implies* $\exists\varphi'' \in [\![\varphi]\!]_X$ *such that* $(\varphi' \sqsubseteq^X_\varphi\ \varphi'')$ *and* $(Q \models\varphi'')$.

Proof: A complete proof can be written using induction on size of P,Q and C. Here we consider one case and the other cases are similar. We restrict our attention to the omission fault case.

If φ is of the form $\langle\mu\rangle\varphi_1$ then there exists C′ such that $(C \xrightarrow{\mu} C')$ and $(C' \models \varphi_1)$. As $\varphi' \in [\![\ \varphi]\!]_O$ it is either of the form $\langle\mu\rangle\varphi'_1$ for φ'_1 in $[\![\ \varphi_1\ ]\!]_O$ or φ' in $[\![\ \varphi_1\ ]\!]_O$. The behaviour of P can be divided into two cases. Either there exists P′ such that $P \xrightarrow{\mu} P'$ or there is no μ move.

If P has an μ move, then φ' is of the first form. Furthermore from the definition of $\prec^O_C$, there exists Q′ such that $Q \xrightarrow{\mu} Q'$. This implies that φ'' can be of the form $\langle\mu\rangle\varphi''_1$ and by induction hypothesis $(\varphi'_1 \sqsubseteq^O_{\varphi_1}\ \varphi''_1)$ and $(Q' \models\varphi''_1)$.

If P has no μ move, then φ' is of the second form. Now, either there exists Q′ such that $Q \xrightarrow{\mu} Q'$ and $(P \prec^O_{C'} Q')$ or $(P \prec^O_{C'} Q)$. If Q had an μ move, then φ'' would be of the form $\langle\mu\rangle\varphi''_1$ and by induction hypothesis $(\varphi' \sqsubseteq^O_{\varphi_1}\ \varphi''_1)$ and $(Q' \models\varphi''_1)$. If Q had no μ move then by induction hypothesis $(\varphi' \sqsubseteq^O_{\varphi_1}\ \varphi'')$. □

The following example illustrates the theorem.

Example 6 *Let* $P = \mu_1 \cdot \mu_2 \cdot \mu_3 \cdot \mathbf{0}$, $Q = \mu_1 \cdot \mu_2 \cdot \mu_4 \cdot \mathbf{0}$ *and* $C = \mu_1 \cdot \mu_5 \cdot \mu_4 \cdot \mathbf{0}$.

Let φ_p *be* $\langle\mu_1\rangle\langle\mu_2\rangle\langle\mu_3\rangle$ *True,* φ_q *be* $\langle\mu_1\rangle\langle\mu_2\rangle\langle\mu_4\rangle$ *True and* φ_c *be* $\langle\mu_1\rangle\langle\mu_5\rangle\langle\mu_4\rangle$ *True.*

It is easy to verify that P satisfies φ_p, *Q satisfies* φ_q *and C satisfies* φ_c.

$P \prec^V_C Q$ *as after exhibiting* μ_1, *P and Q can exhibit* μ_2 *which is a garbled version of* μ_5 *after which C and Q agree on* μ_4 *while P continues to be garbled and exhibits* μ_3.

We now show $\varphi_p \sqsubseteq^V_{\varphi_c} \varphi_q$. *The first clause of definition 17 states that* $True \sqsubseteq^V_{True} True$. *Hence from definition 16 and the second clause of definition 17,* $\langle\mu_3\rangle$ *True is an element of* $[\![\ \langle\mu_4\rangle$ *True* $]\!]_V$ *and* $\langle\mu_3\rangle$ *True* $\sqsubseteq^V_{\langle\mu_4\rangle True}\ \langle\mu_4\rangle$ *True.*

By continuing the above argument it can be shown that φ_p *and* φ_q *are elements of* $[\![\ \varphi_c\ ]\!]_V$ *and that* $\varphi_p \sqsubseteq^V_{\varphi_c} \varphi_q$.

5 Conclusion and Future Work

In this paper we have presented a simple syntax and operational semantics for replicated systems. We have considered three types of faults and defined preorders induced by them. These preorders were indexed by a correctness criteria. P $\prec_C^T$ Q indicates that Q is no more faulty than P for faults of type T given correctness criteria C. We have also defined fault introduction and presented a few preliminary results relating the fault preorders and fault-introduction. We have presented a modal logic characterisation of the fault-introduction and fault preorders.

The main issues that need further investigation include applying the technique to other types of fault tolerant systems, considering recursive processes and communication and extending the modal characterisation to the full modal-μ calculus.

Synchronous majority voting is only one technique to attain fault tolerance. As replication of sub-systems can be expensive, other techniques such as resourceful systems [1] are also used. The applicability of this work to other techniques needs further investigation.

In this paper we have not considered recursion. The main issue in fault-tolerant recursive systems is whether subsequent unfoldings are the modified faulty process or the original process. This depends of whether the fault is considered permanent or transient. Also many of our results depend on finite behaviour. It remains to be seen if they can be generalised to regular or context free behaviours.

The reason for excluding communication is that the effect of a fault on complementary and hidden actions needs to be considered. Consider, for example, the CCS process (say P) $(\mu\cdot\mathrm{Q} \mid \overline{\mu}\cdot\mathrm{R}) \setminus \{\mu\}$ and its behaviour under an μ-omission fault. If the μ-omission fault also omits $\overline{\mu}$, P is weakly bisimilar to (Q | R), i.e., P under fault. However, there is no reason to believe that faults will be 'well-behaved'. If an μ-omission does not affect $\overline{\mu}$, then P affected by the fault is related to Q. While this is acceptable, the process $(\mu_1\cdot\mathrm{Q} \mid \overline{\mu_1}\cdot\mathrm{R}) \setminus \{\mu_1\}$ is not affected by an μ-omission. Therefore, the choice of local names has an impact on the fault semantics. One could argue that a fault should not affect hidden actions but such an assumption would not be realistic.

In the modal characterisation we did not consider explicit negation nor the necessity ($[\mu]$) modality. The reason is that we have been unable to provide reasonable transformations that characterise fault-tolerance. For example, consider the formula $[\mu_2]$False and a process (say P) $\mu_1 \cdot \mu_2$. While P satisfies the formula, P under an μ_1 omission does not satisfy the impossibility requirement. In this particular case the faulty process will satisfy $\langle\mu_2\rangle$True. While one could translate $[\mu_2]$False to $[\mu_2]\mathrm{False} \vee \langle\mu_2\rangle\mathrm{True}$, the translation is not very meaningful as it is equivalent to True. A general non-trivial scheme to translate and impose an order on modal formulae involving negation is under investigation.

Acknowledgements

The author acknowledges the many helpful comments from the referees. This research has been partially supported by University of Canterbury Grant No 1787123.

References

[1] R. J. Abbot. Resourceful Systems for Fault Tolerance, Reliability and Safety. *ACM Computing Surveys*, 22(1), 1990.

[2] L. Aceto and M. Hennessy. Adding Action Refinement to a Finite Process Algebra. In *ICALP -91, LNCS 510*. Springer Verlag, 1991.

[3] A. Avizienis. The N-version Approach to Fault-Tolerant Software. *IEEE Transactions on Software Engineering*, 11(12):1491–1501, Dec 1985.

[4] J. A. Bergstra and J. W. Klop. Process Theory Based on Bisimulation Semantics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354*, pages 50–122. Springer Verlag, 1988.

[5] F. Cristian. A Rigorous Approach to Fault-Tolerant Programming. *IEEE Transactions on Software Engineering*, 11(1):23–31, 1985.

[6] F. Cristian. Understanding Fault-Tolerant Distributed Systems. *Communications of the ACM*, 34(2):56–78, February 1991.

[7] R. Gorrieri, S. Marchetti, and U. Montanari. A2CCS: A Simple Extension of CCS for Handling Atomic Actions. In *13th Colloquim on Trees in Algebra and Programming*, pages 258–270. Springer Verlag, 1988.

[8] R. E. Harper and J. H. Lala. Fault-tolerant Parallel Processor. *AIAA Journal of Guidance, Control and Dynamics*, 14(3):554–563, May-June 1991.

[9] M. C. B. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.

[10] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1985.

[11] M. Koutny, L. Mancini, and G. Pappalardo. Formalising Replicated Distributed Processing. In *Proceedings of the 10th Symposium on Reliable Distributed Systems*, pages 108–117, Pisa, Italy, 1991. IEEE.

[12] P. Krishnan. Distributed CCS. In *Theories of Concurrency: Unification and Extension: CONCUR-91, LNCS:527*, pages 393–407. Springer-Verlag, August 1991.

[13] P. Krishnan. A Semantics for Multiprocessor Systems. In *European Symposium On Programming (ESOP): LNCS 582*, pages 307–320, Rennes, France, February 1992. Springer-Verlag.

[14] P. Krishnan. Pre-Orders for Fault-Tolerance. In *Proceedings of the 16th Australian Computer Science Conference*, pages 693–701, Brisbane, 1993.

[15] P. Krishnan and B. J. McKenzie. A Process Algebraic Approach to Fault-Tolerance. In *Proceedings of the 15th Australian Computer Science Conference*, pages 473–485, Hobart, 1992.

[16] K. G. Larsen. A Context Dependent Equivalence Between Processes. *Theoretical Computer Science*, 49:185–215, 1987.

[17] K. G. Larsen. Modal specifications. In *Proceedings of the Workshop in Automatic Verification Methods for Finite-State Systems: LNCS 407*, pages 232–246. Springer Verlag, 1989.

[18] R. Milner. *Communication and Concurrency.* Prentice Hall International, 1989.

[19] M. Nielsen, U. Engberg, and K. S. Larsen. Fully Abstract Models for a Process Language with Refinement. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354*, pages 523–548. Springer Verlag, 1989.

[20] D. Park. Concurrency and Automata on Infinite Sequences. In *Proceedings of the 5th GI Conference, LNCS-104.* Springer Verlag, 1981.

[21] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[22] A. Pnueli. Linear and Branching Structures in the Semantics and Logics of Reactive Systems. In *ICALP -85: LNCS 194*, pages 15–32. Springer Verlag, 1985.

[23] C. Stirling. An Introduction to Modal and Temporal Logics for CCS. In *Joint UK/Japan Workshop on Concurrency:LNCS 491*, pages 2–20, 1989.